Particle filter for platoon based models of urban traffic

NICOLAE MARINICĂ
Ghent University
Faculty of Engineering
Technologiepark 913, 9052 Zwijnaarde
BELGIUM
NicolaeEmanuel.Marinica@ugent.be

RENÉ BOEL
Ghent University
Faculty of Engineering
Technologiepark 913, 9052 Zwijnaarde
BELGIUM
Rene.Boel@UGent.be

Abstract: This paper proposes a particle filter (PF) state estimator, using a platoon based model for urban traffic networks. The urban traffic network model consists of signalized intersections (representing queues of vehicles competing for service) connected to each other through links with predefined receiving capacities and stochastic delays. Sensors detect the passage of vehicles at the sensor locations. The algorithm is flexible and robust and can be used in real-time applications such as on-line control of switching times of traffic lights.

Key–Words: Bayesian estimation, particle filtering, urban traffic, platoon based model, stochastic systems.

1 Introduction

The estimation and prediction of the traffic state in an urban traffic network is a very important part of the feedback loop that will in the future implement on-line road traffic management, ensuring efficient and safe operations. Many different models for both freeway traffic and for urban traffic have been developed, each with a different representation of the traffic state. Urban traffic, the topic of this paper, can be described by macroscopic models [5], [6] representing the average traffic behavior in terms of the aggregated variables density and speed, as measured at different locations, by microscopic models that represent the behavior of each vehicle separately or by hybrid models, such as the platoon based model that we are using in this paper.

Platoon based models group vehicles in platoons that travel at approximately the same speed, closely following each other. The state of the traffic network at time t is represented in such a platoon based model by the location at time t of the head of each platoon and by its size. Queues behind traffic lights can be interpreted also as stationary platoons. The choice for a platoon based model is dictated by the fact that the traffic data we have available, covering measurements over more than 40 days in a network with 5 signalized intersections, several roundabouts, and a number of additional unsignalized intersections, indicate that vehicles travel most of the time in platoons. Moreover the purpose of the estimation algorithms developed in this paper is to enable a feedback controller that coordinates the switching times of the traffic lights so as to minimize the delay. It is intuitively clear (and it will be shown in detail in a forthcoming paper [17]) that future switching times should depend on the expected arrival times of these platoons. For this reason we try to develop recursive estimators for the location and size of the platoons in the network, and for the size of the queues of waiting vehicles.

The computational complexity of Bayesian recursive state estimators is prohibitive for large systems, like models of traffic networks, because it requires calculation of conditional densities of the current state. Recently particle filtering (PF) has been proposed as a method for approximating this conditional density by an empirical histogram obtained via Monte Carlo simulation. There is already a rich literature about PF for traffic estimation for motorway traffic [2], [3], [16] and [4], but little about state estimation for urban traffic. Other approximate methods for recursive traffic state estimation apply the Extended Kalman Filter (EKF) to macroscopic models [7] [13] like the freeway traffic flow model METANET [8]. This approach has the disadvantage of the EKF technique to rely on a linearization of the state and measurements models, which can cause stability problems. Moreover the linearization is not applicable to discrete event models like platoon based models.

The PF runs N simulations of the platoon based model of the urban traffic over the time interval $[0, t]$. That is why the PF requires only a computationally efficient model used for the evolution of the traffic state. The platoon based model used in this paper can indeed be implemented very efficiently as a discrete event system (DES) simulator, because it needs to keep track only in its event list of the arrival times at a few locations (e.g. upstream of an intersection)

of the successive platoons, and of their sizes. Since the number of events to be executed by the simulator is proportional to the number of platoons, much smaller than the number of vehicles especially under congested traffic conditions, this significantly reduces the simulation time. Moreover the number of the platoons remains approximately independent of traffic intensity since the expected size of the platoons grows with the traffic intensity but the arrival rate of the platoons remains approximately constant, as shown by the data set (which actually provides us with the times at which an axle of a vehicles crosses the location of a sensor, where we have about 100 sensor locations in the network under consideration) to be presented in the forthcoming paper [18].

The platoon based model is presented in section 2. Section 3 contains a technical description of the PF for a platoon based model. The stochastic simulation setup of the particles is tackled in section 4. The last two sections are dedicated to some examples showing the feasibility and the robustness of the approach followed by conclusions.

2 Model

The topology of the urban traffic network consists of *links, intersections, and sensor locations*. The *state* x_t of the model at time t represents the location and the size of all the platoons in the system at time t, including the queues stopped at the intersections. The state variables expressing the location and size of a platoon will be defined below. The state x_t also includes the mode (red, green or yellow for each traffic light) of all traffic lights at time t. Before defining variables describing a platoon location we define the different types of components that are considered in an urban traffic network:

- An *intersection* has as many access points as there are preselection lanes for traffic arriving at the physical intersection, and moreover has (at most) 4 exit points. Provided the priority and safety rules (mode of traffic lights, or other priority rules, including special rules for preselection lanes for left or right turning traffic) are satisfied a platoon waiting in a preselection lane moves from an entrance point of the intersection to the exit point corresponding to the preselection lane. This exit point is the entrance point of a downstream link. A platoon that is allowed to leave a preselection lane immediately crosses the sensor at the entrance point of the intersection, and after a random delay $\Delta_{intersection}$ with predefined mean value, crosses the sensor location at the entrance of the downstream link it is moving to.
- The head of a platoon traveling through a *link* moves from the entrance point of the link to the exit point of the link with a random delay (corresponding to the average speed along the road represented by that link, and by the standard deviation of that speed; accidents can be modeled by large random increases in this delay). The size of the platoon determines the difference between the time when the head of the platoon reaches the exit point and the time when the tail reaches this exit point. A link has a predefined storage capacity (modeling spill-backs). Platoons merge when the head of one platoon catches up the tail of another platoon, but in our current implementation this is checked only when the platoons they reach the end of the queues formed in front of the traffic light of the downstream intersection.
- The model requires at least a *sensor locations* at each entrance and exit point of a component, but more sensor locations in the middle of a link can be included in the model. The sensors at the entrance, resp. the exit point of component c are denoted further on as k_c^{in}, resp. k_c^{out}. At sensor location ℓ_j the model generates as output a list of all times when an axle of a vehicle passed the sensor location. Currently this list is translated into a noisy sequence of numbers of vehicles $Y_j(t_k) = h(x(t_k)) + v_{k,j}$ that are detected in the time interval $[k \cdot \Delta t\,,\ (k+1) \cdot \Delta t]$ seconds. As shown in figure 2, knowing the state $x(t_k)$ at time t_k, and thus knowing the location of head and tail, and the size, of each platoon crossing the sensor location ℓ_j, allows the calculation of the number $h(x(t_k))$ of vehicles that cross sensor location ℓ_j in the interval $[k \cdot \Delta t\,,\ (k+1) \cdot \Delta t]$. The noisy output $Y_j(t_k)$ adds to $h(x(t_k))$ the noise term $v_j(k)$ with a known discrete random distribution.

The platoon based model of urban traffic, as used in this paper, is a Discrete Event Systems (DES) model that describes the progress of platoons of vehicles traveling through the components of the network. Physically, a *platoon* represents a formation of vehicles closely following each other with approximately the same speed. Vehicles that are waiting at an intersection (behind a red light, or because the intersection is blocked for other reasons) are considered as a stationary platoon. The set of platoons $P_i, i \in I(t)$ of platoons that is present in the network at time t is defined by the index set $I(t) \subset N_0$. Thus the state $\{n_i(t)\,, \ell_i(t)\,, \tau_{\ell_i(t),i,head}(t)\,, tau_{\ell_i(t),i,tail}(t)\,, i \in I(t),$ *mode of each traffic light* $\} = x(t)$ is the state of the system at time t. The location and the size of platoon $P_i, i \in I(t)$ at time t are characterized by:

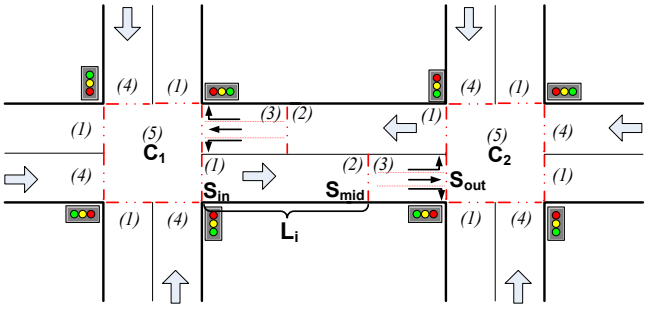

Figure 1: The road topology and the sensors

$n_i(t)$ number of vehicles in platoon i, $\ell_i(t)$ - last sensor location passed by the head of the platoon i, and $\tau_{\ell_i(t),i,head}(t)$, $\tau_{\ell_i(t),i,tail}(t)$ times at which head and tail of the platoon i pass (or will pass, in case the platoon is still covering the sensor location) the sensor location $\ell_i(t)$. The network is covered by a sufficient number of sensors $\ell = 1, \ldots, L$, so that detecting these passage times of platoons at the sensor locations provides sufficiently detailed information on the state of system in order to describe the future evolution, and in order to implement a state feedback control law. The location of the platoons is described by the times $\tau_{\ell_i(t),i,head}(t)$, at which the head or $\tau_{\ell_i(t),i,tail}(t)$ when the tail of a platoon passes a sensor location. Events in this DES system occur each

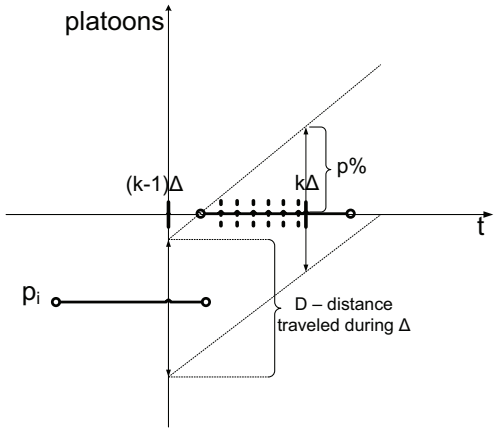

Figure 2: Sensor S_j detecting a platoon during Δt

time when the head or the tail of a platoon crosses a sensor location, and each time a traffic lights changes its mode. Knowing the current state $x(T_n)$ at the time T_n when the n-th event occurs it is possible to calculate the time of the next events to be added to the event list. If the event is the arrival time $\tau_{\ell_i(t),i,head}(t)$ of the head of a platoon P_i at sensor location $\ell_i(t)$, where $\ell_i(t) = k^{in}_{link_n}$ corresponds to the entrance of a $link_n$, then the random travel time $\Delta_{n,i}$ for a platoon to travel through link n will be generated, and the event corresponding to the the head of platoon P_i crossing the sensor location $k^{out}_{link_n}$ at the exit of the same link is given by $\tau_{link^{out}_n,i,head} = \tau_{link^{out}_n,i,head} + \Delta_{link_n,i}$. Similarly the crossing of the head of a platoon at time $\tau_{k^{in}_{intersection_n},i,head}$ of the entrance sensor location of an intersection generates a random time $\tau_{link^{out}_n,i,head}$, which is $\Delta_{intersection}$ time units later, the transition that the tail of platoon P_i crosses the entrance sensor location of the downstream link of this intersection. Similar rules apply for calculating the time at which the events occur when the tail of a platoon crosses a sensor location.

Further events are defined by the arrival of the head, or of the tail, of a new platoon that enters the network at a sensor location at the boundary of the network. The events corresponding to the switching of traffic lights have known timings and can therefore also be included easily in the DES model.

After the occurrence of an event at time T_n the state $x(T_n-)$ must be updated to $x(T_n+)$ in order to take into account the type of event that took place. If this event is an arrival of a new platoon, or the departure of a platoon that entered or left the system through a boundary location then one must update $I(t)$ appropriately, and add or delete the values $\{n_i(t), \ell_i(t), \tau_{\ell_i(t),i,head}(t), \tau_{\ell_i(t),i,tail}(t)\}$ of the added, resp. deleted platoon. If the event is the crossing of the head of platoon i of sensor location ℓ_i then the last sensor location for platoon $i \in I(t)$ must be updated to the new value, together with the corresponding $\tau_{\ell_i(t),i,head}(t), \tau_{\ell_i(t),i,tail}(t)$.

In order to limit the size of this paper we do not describe in detail how the state is updated when a platoon splits up in several smaller platoons, either due to random effects, or when the platoon arrives at the end of a link and has the choice between more than one preselection lanes. We assume that the vehicles of each platoon are randomly routed at intersection (with probabilities determined by origin-destination flow rates). This can be implemented fairly straightforwardly in the simulations tool.

The current implementation is a single class model where all vehicles are identical, having 2 axles (remember that our sensors measure axles passing sensor locations).

The particles to be used by the particle filter (PF) below, as described in the next section, will be generated by using a simulation tool that implements the model described in this section.

3 Technical description of Particle Filter

The recursive Bayesian estimation algorithm evaluates of the posterior probability distribution of the state vector. Since this is computationally infeasible one often replaces this by a particle filter, where an empirical histogram obtained via Monte Carlo simulation of N particles approximates this conditional probability distribution [14] [15]. In this case study we generate the particles according to the model described in section 2 and use the outcome of the particle filter to estimate the size and location of the platoons in the network, and more particularly in order to estimate the size of the queues formed on the link upstream of intersection C_2. The challenges are represented by the fact that the model of the queues is a pure integrator summing all noise terms without a smart correction algorithm. The estimation error decreases via the information on empty queues (the outflow at $S_{out} \leq \mu_{min}$ where μ expresses the arrival rate) and full queues (fill up space between S_{mid} and S_{in}). An advantage of our model is that we can use the raw measured data, the successive time instants when a sensor detects the passage of an axle, in order to estimate location and size of platoons, and even queue sizes, without requiring any preprocessing of the data.

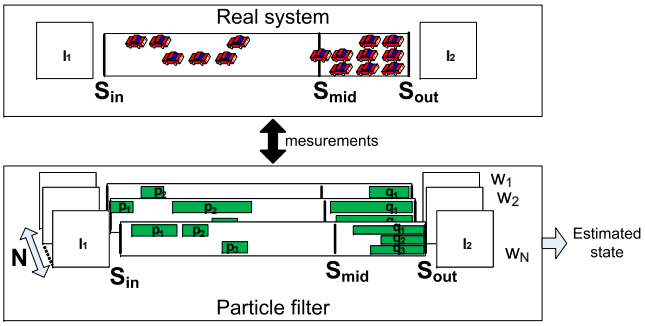

Figure 3: Practical representation of the PF in the real system

Recursive Bayesian estimation applies Bayes' rule in order to update the posterior probability density function (PDF) $p\left(x_k|Z^k\right)$ (where $x_k = x(t_k)$, $t_k = k \cdot \Delta t$) of the state vector x_k at time instant t_k, given all the sensor measurements $Z^k = \{z_1, ..., z_k\}$ available at time t_k. By applying Bayes' rule we can write the conditional density function $p\left(x_k|Z^k\right)$ of the state x_k given the PDF $p\left(x_k|Z^{k-1}\right)$ and given the new observation in $[t_{k-1}, t_k)$ as follows:

$$p\left(x_k|Z^k\right) = \frac{p\left(z_k|x_k\right)p\left(x_k|Z^{k-1}\right)}{p\left(z_k|Z^{k-1}\right)} \qquad (1)$$

where $p(x_k|Z^{k-1})$ is obtained from $p(x_{k-1}|Z^{k-1})$ by implementing the state update model:

$$p\left(z_k, Z^{k-1}\right) = \int_{\mathbb{R}^{n_x}} p\left(x_k|x_{k-1}\right) p\left(x_{k-1}|Z^{k-1}\right) dx_{k-1} \qquad (2)$$

Equation (1)is called the measurement update step, while equation (2) is called the state update prediction step. The conditional distribution $p\left(x_k|Z^k\right)$ expresses how likely it is that x_k is an explanation of the observed measurements $Z^k = \{z_1, ..., z_k\}$.

The evaluation of the denominator in the measurement update step is computationally expensive especially when the state space is too large (e.g. like in case of the state space x_k of a traffic network), because it requires integration over a large and complicated state space. The PF avoids this integration by replacing it by the average over a fixed number N of samples, called particles further on. This makes the problem scalable. These particles can be generated using N independent Monte Carlo simulations running in parallel on a computer, each simulations run $i = 1, \ldots, N$ generating a state trajectory $x_\ell^i, \ell = 1, \ldots, k$, implementing the dynamical model of the plant. In our case study this is achieved by implementing the model described in section 2. A weight w_k^i is associated to each particle, expressing at time t_k how well this particle i corresponds to the observations Z^k, available at moment t_k. In the practical implementation, the sum of the weights must be normalized to 1 for numerical reasons. The PF method approximates $p\left(x_k|Z^k\right)$ by the empirical histogram corresponding to these N *particles* and their weights.

In prac tice the PF is implemented recursively as described in Algorithm 1. Initially N random selections of sizes and locations of platoons are generated as shown in figure 3, are and assigned the weights $1/N$ each. From then on one implements for $k = 0, 1, \ldots,$ the following recursion. Generate for each particle the state evolution in the interval $[k.\Delta t, (k+1).\Delta t)$ according to the model of the plant

(see line 7 in algorithm 1; in our case study this is the platoon based model described in section 2). After the arrival of a new update of the observation vector, for example Z_{k+1} at moment $(k+1)\cdot\Delta t$, the PF will update the weight $w_k^{(i)}$ of the i-th particle by multiplying it with $p\left(z_k|x_k^i\right)$ (see line 10 in algorithm 1). In our case study this likelihood $p\left(z_k|x_k^i\right)$ of the observed counts of axles will be obtained from the model of the sensors. For sufficiently large N the particles will generate an empirical histogram that approximates the true PDF of the state sufficiently accurately. The algorithm 1 describes the implementation of the particle filter.

Algorithm 1 Particle filter algorithm

Ensure: $\rightarrow$ Initialization of the particles $k=0$
1: **for** $i=1$ to N **do**
2: generate the i^{th} sample $\left\{x_0^i\right\}$
3: initialize i^{th} weight with $w_0^i = 1/N$
4: **end for**
5: **for all** k such that $k \geq 1$ **do**
Ensure: $\rightarrow$ Prediction step
6: **for** $i=1$ to N **do**
7: update the i^{th} sample $\left\{x_k^i\right\}$ according to $p\left(x_k|x_{k-1}^i\right)$
8: **end for**
Ensure: $\rightarrow$ Measurement step
9: **for** $i=1$ to N **do**
10: update the weight of the i^{th} sample $w_k^i = w_{k-1}^i \cdot p\left(z_k|x_k^i\right)$
11: **end for**
Ensure: $\rightarrow$ Normalization
12: **for** $i=1$ to N **do**
13: normalize the weight of the i^{th} sample $\hat{w_k^i} = {}^{w_k^i}/_{\sum_{i=1}^N w_k^i}$
14: **end for**
Ensure: $\rightarrow$ Output
15: order particles by the size of the weight.
16: $\hat{x}_s \Leftarrow$ particle with the highest weight.
Ensure: $\rightarrow$ Resampling
17: **for** $i=1$ to $N/2$ **do**
18: for pair $\left(P_k^i, P_k^{N-i+1}\right)$
19: replace the state of the particleP_k^{N-i+1} with the state of the particle P_k^i
20: $w_k^{i_n ew} = \left(w_k^i + w_k^{N-i+1}\right)/2$
21: $w_k^i = w_k^{i_n ew}$ and $w_k^{N-i+1} = w_k^{i_n ew}$
22: **end for**
Ensure: $\rightarrow$ Time step update
23: $k \Leftarrow k+\Delta$
24: **end for**

There is a practical problem that can appear after a few time steps. The set of particles with significant weight may no longer be sufficiently diverse to properly approximate an empirical histogram. Only a few particles have significant weights, others have become so unlikely that their weight. It means that some weights may become so small that it seems useless to maintain these particles in the set of samples. There are many resampling techniques described in literature as [9], [10] that try to find a good compromise between keeping all the useful information (an unlikely particle may become likely after the next observation) and avoiding time consuming calculations with very unlikely particles. The resampling method used in this paper is described on lines 17-22 of algorithm 1: it duplicates the $N/2$ most likely particles, and throws away the $N/2$ least likely particles.

4 Stochastic simulation of particles

The model of section 2 is translated into a discrete event simulations tool, that generates the evolution of the particles in the intervals $[k.\delta t, (k+1).\delta t)$, as shown in line 7 of algorithm 1. The agenda of this DES simulator is stored as a list sorted according to increasing event occurrence times. It keeps track of all the future events for which the occurrence time has already been calculated. The system starts executing with an initially selected value for the state at time $t=0$, and an agenda of future events that is compatible with this initial state (lines 3 and 4 of algorithm 1. The system state and the agenda are then updated recursively by selecting the first event time T_n in the agenda, calculating the corresponding state update $x(T_n+)$, deleting the event that has just been executed at T_n from the agenda, and adding all the new events (with event time $> T_n$) to the agenda whose occurrence can be calculated as a result of the occurrence of the event at time T_n.

The execution of an event like the crossing of a sensor location by the head of a platoon generates new events with a random delay, sampled by using a random number generator implementing the selected probability distributions of $\Delta_{intersection}$ and of Δ_{link}. In the implementation used below the time delay along a link k is calculated by dividing the length L_k of the link k by a a random speed $r.V_{max,k}$ where $r = 1,\ 0.9$, resp. 0.8 with probabilities $0.8,\ 0.15$, resp. 0.05.

In order to use the simulator for carrying out experiments with the particle filter described in section 3 we also need to simulate the sensor output $Y_\ell(t_k), k = 0, 1, \ldots$. This is generated as follows. First we assume that each of the $n_i(t)$ axles of platoon

$P_i, i \in I(t)$ that is crossing sensor location $\ell(t_k)$ at time t_k generates a pulse counted by the sensor with probability P_{detect}. The total number $Z_{\ell,i}$ of axles that will be detected by sensor ℓ in the interval $[\tau_{\ell_i(t),i,head}(t), \tau_{\ell_i(t),i,tail}(t)]$ is obtained as a binomially distributed random variable with parameters $(n_i(t), p)$. The $Z_{\ell,i}$ pulses are then allocated to the measurement at time t_k proportional to the length of the overlap between the intervals $[k.\Delta t, (k+1).\Delta t)$ and $[\tau_{\ell_i(t),i,head}(t), \tau_{\ell_i(t),i,tail}(t)]$, i.e. $\hat{Y}_\ell(t_k) = \Sigma_{i\in I(t)} \frac{|[k.\Delta t,(k+1).\Delta t) \bigcap [\tau_{\ell_i(t),i,head}(t),\tau_{\ell_i(t),i,tail}(t)]|.Z_{\ell,i}}{|[\tau_{\ell_i(t),i,head}(t),\tau_{\ell_i(t),i,tail}(t)]|}$. The actual output $Y_\ell(t_k)$ is obtained by adding to $\hat{Y}_\ell(t_k)$ a random number $v_{false,\ell,n}$ of false detections, with a selected distribution for $v_{false,\ell,n}$ (in the experiments we use $\mathcal{P}(v_{false,\ell,n} = j) = p_j$ with $p_0 = 0.95$, $p_1 = 0.04$, $p_2 = 0.01$. The model incorporates further randomness in the evolution of the particles as follows:

- randomly selected speed to determine the event time for the arrival of the platoon at the next sensor location
- for each platoon that is entering on a link a number of cars is added or subtracted from that platoon in order to model the cars that are leaving the link (e.g. entering a parking lot) or joining the platoon by entering on the link (e.g. leaving a parking lot). The current model does not keep track of the number of parked cars.
- splitting of platoons at intersections according to a multinomial distributions.
- generating new platoons with uniformly distributed number of cars and exponentially distributed interarrival times (gaps) between successive platoons.

The practical implementation was done in Matlab. Because the parallel simulation of the N particles cannot be performed in a real parallel manner we had to run each particle for Δt time units in a single thread of execution. The only condition to be able to run the PF in real time is the $running\_time_{(N\ particles)} < \Delta t$ the update time of the measurements.

5 Examples of platoon based filter

Our data sets of real traffic, used in developing the platoon based model, do not provide accurate information on queue sizes. Therefore we validated the particle filter by comparing the estimates obtained from the PF with synthetic data, called the "ground truth model". The ground truth model generates a state trajectory, with $\Delta t = 3sec$, according to a platoon based model. The observations used as input for the PF are obtained by letting the platoons of the ground truth model interact with the sensors, according the model described in section 2.

We give here some examples obtained with the described implementation of the PF working with the platoon based model. The number of particles used in the examples is 202. The estimated state shown in figures $4, 6$, and 8 has fast variations around the real values of the queues due to the fact that we the queue size that is shown as output is the most likely queue size, generated by the the particle that has the highest weight at $k.\Delta t$. Showing histograms would be too difficult to interpret easily, while time averages over all particles smooth out some interesting details. After each measurement update the weights are updated and a different particle will become the most likely particle, explaining the fast variations in queue size estimates shown in figs. $4, 6$, and 8. The goal of this work is to use the $K(< N)$ most likely particles in a feedback controller for the switching times of traffic lights. For a reasonable number K of particles the variations shown in figs. $4, 6$, and 8 will not have too much influence on this feedback controller.

1. The first example presents the case where there is no difference between the model used as ground truth and the model used for the particles. The differences between ground state and particle evolution is due only to the noise in the model. Sufficient diversity is assured by using $N = 202$ particles. Figure 5 shows the evolution of the weights of the particles in a 4.15 minutes interval while figure 4 shows in red full lines the evolution of the real queue size, and in blue dashed lines the evolution of the estimated queue size (the same color codes are used in figs. 6 and 8). Computation time was 31.08[s], i.e. 8 times faster than real time.
2. In the second example the model used for generating the synthetic data for the ground truth uses $15^{'}$ as average interplatoon time, while the PF uses $10^{'}$ for this interplatoon time. In other words the PF thinks the traffic is 50% busier than it in reality is. The PF is recovering quickly and it follows approximately the trajectory of the real system, showing robustness of the PF against errors in the model parameters. The real time was 5.15[min] and the computation time was 39.97[s]. The figure 7 shows the evolution of the weights of the particles while figure 6 shows in red the evolution of the real state and in blue dashed line the evolution of the estimated state.

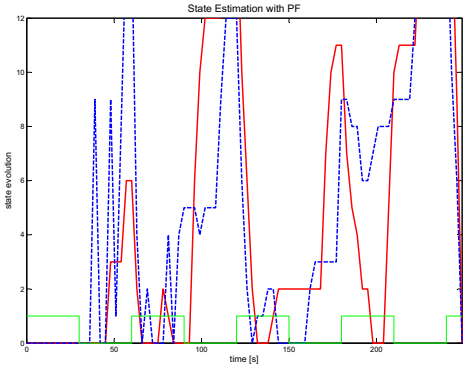

Figure 4: Real queues (red line) vs. estimated queues (blue dashed line)

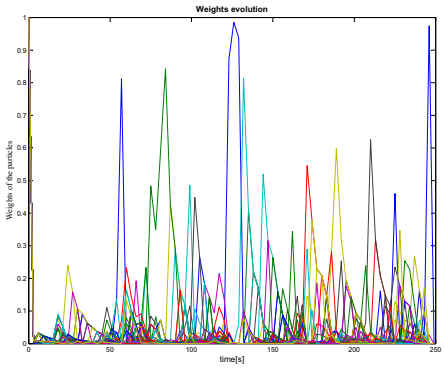

Figure 5: Weights evolution of the particles

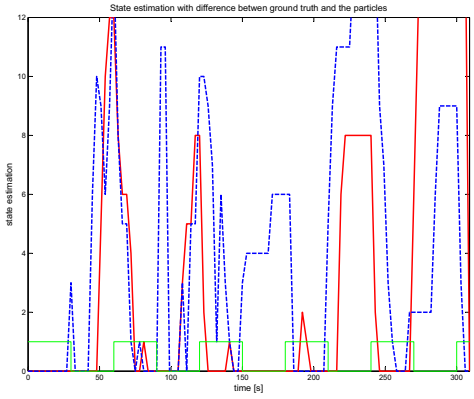

Figure 6: Real queues (red line) vs. estimated queues (blue dashed line) with difference between models

3. The third example illustrates the robustness of the PF with the platoon based model with respect to modelling errors like delays of vehicles. The model used for generating the synthetic data for the ground truth has the maximum speed $V_{max,1}$ for the platoons that enters a link set to 30 km/h while for the model used for the particles this speed is $V_{max} = 60km/h$. The most likely queue size shown in fig. 8 is quickly following the trajectory of the real system with a small error. The real time was 5.05[min] and the computation time was 41.32[s].

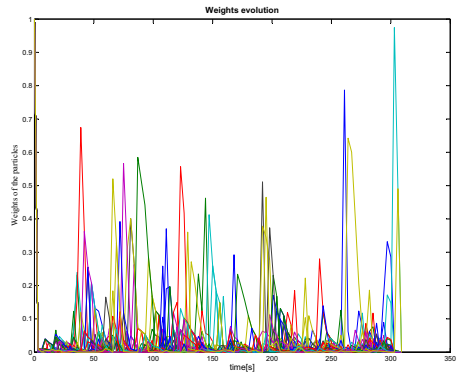

Figure 7: Weights evolution with difference between models

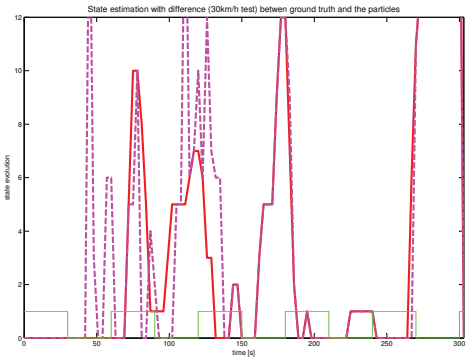

Figure 8: Real queues (red line) vs. estimated queues (blue dashed line) with difference between models

6 Conclusions and future work

We introduced a discrete event system model for urban traffic based on platoons of vehicles. Validation and parameter tuning of this platoon based model uses a large set of data collected along the N47 between Lokeren and Dendermonde, in Belgium. This analysis will be reported in a forthcoming paper. The platoon based model is then applied in a PF used for state estimation. We presented the algorithm used for the implementation of the PF and we also gave insights into the real implementation. The PF is able to cope with

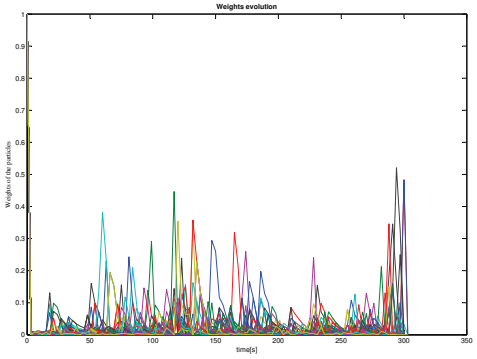

Figure 9: Weights evolution with difference between models

different uncertainties (in the data, as well as model uncertainty) and allows for information fusing from different measurement sources.

We have shown that the PF used with the platoon based model is fast enough for small networks, but for large networks the number of events will grow linearly. Therefore for large networks a distributed implementation of the PF, like for the free flow traffic as presented in [11] and [12], can be used. The number of particles can also be further increased. We have to extend the robustness analysis against model changes e.g. accidents. Incident detection can be performed as a further analysis of the output of the PF. The future work is also related to the extension of the current model. We are investigating the possibility of model predictive feedback control type paradigms, where the switching times of the traffic lights will be selected so as to guarantee acceptable performance for all the particles with a weight above a chosen threshold.

Acknowledgements: This paper presents research results of the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office and EUFP7 project CON4COORD. The scientific responsibility rests with its authors.

References:

[1] D. Helbing, Traffic and related self-driven many-particle systems, *Rev.Modern Phys.*, 73:1067-1141, 2002.

[2] R. Boel and L. Mihaylova, A compositional stochastic model for real-time freeway traffic simulation, *Transportation Research B*, vol.40, no.4, 319-334, 2006.

[3] L. Mihaylova and R. Boel, A particle filter for freeway traffic estimation, *Proc.43rd IEEE Conf. on Dec. & Contr.*, vol.40, no.4, 319-334, 2004.

[4] L. Mihaylova, R. Boel and A. Hegyi, Freeway traffic estimation within particle filtering framework, *Automatica*, vol.43, no.2, 290-300, 2006.

[5] M.van den Berg,A. Hegyi,B. De Schutter,and J. Hellendoorn, A macroscopic traffic flow model for integrated control of freeway and urban traffic networks, *Proc.42nd IEEE Conf. Dec. and Control*,Maui,Hawaii, pp.2774 2779, 2003.

[6] K. Wen, S. Qu, Y. Zhang, A Control-oriented Macroscopic Traffic Flow Model for Urban Diverse Intersections, *2009 International Asia Conference on Informatics in Control, Automation and Robotics*,pp. 62-66, 2009.

[7] Y. Wang, M. Papageorgiou, Real-time freeway traffic state estimation based on extended Kalman filter: a general approach., *Transp. Res. B*, 39(2):141-167, 2005.

[8] M. Papageorgiou, J.-M. Blosseville, Macroscopic modelling of traffic flow on the boulevard Périphérique in Paris., *Transp. Res. B*, 23(1):29-47, 1989.

[9] M. Bolic, P.M. Djuric, and S. Hong, Resampling Algorithms for Particle Filters: A Computational Complexity Perspective, *EURASIP Journal on Applied Signal Processing*, no. 15, pp. 2267-2277, 2004

[10] R. Douc, O. Cappe, Comparison of resampling schemes for particle filtering, Image and Signal Processing and Analysis, *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis*, pp. 64 - 69, 2005.

[11] L. Mihaylova, A. Gning, V. Doychinov and R. Boel, Parallelised Gaussian Mixture Filtering for Vehicular Traffic Flow Estimation, *Proceedings INFORMATIK2009*, 09/2009, Luebeck, Germany, 2009.

[12] A. Hegyi, L. Mihaylova, R. Boel, Z. Lendek, Parallelised Particle Filtering for Freeway Traffic State Tracking, Tutorial Session "Motorway Traffic Surveillance and Control II" (TuD15), *European Control Conference ECC07*, Kos, Greece, 2007.

[13] Y. Wang, M. Papageorgiou, and A. Messmer, Motorway traffic state estimation based on extended Kalman filter, *Proceedings of the European Control Conf.*, UK, 2003

[14] A. Doucet, N. Freitas, and Eds.N. Gordon, Sequential Monte Carlo Methods in Practice., *New York: Springer-Verlag*, 2001.

[15] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P-J. Nordlund, Particle filters for positioning, navigation and tracking.,*IEEE Transactions on Signal Processing*, 50(2):425-437, 2002.

[16] X. Sun, L. Muñoz, and R. Horowitz, Highway traffic state estimation using improved mixture Kalman filters for effective ramp metering control., *Proc. of 42th IEEE Conf. on Decision and Control*, pages 6333-6338. , USA, 2003.

[17] N. Marinica and R. Boel, Coordinated on-line control of traffic lights using a platoon based model., In preparation, 2011.

[18] N. Marinica, H. Sutarto, and R. Boel, Statistical analysis and model validation using urban traffic data., In preparation, 2011.